대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2002-0046801
Application Number PATENT-2002-0046801

출 원 년 월 일 : 2002년 08월 08일
Date of Application AUG 08, 2002

출 원 인 : 삼성전자주식회사
Applicant(s) SAMSUNG ELECTRONICS CO., LTD.

2003 년 02 월 06 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0005
【제출일자】	2002.08.08
【국제특허분류】	G11B
【발명의 명칭】	태스크 기반의 비선형 하이퍼비디오 편집방법 및 그 장치
【발명의 영문명칭】	Task oriented nonlinear hypervideo editing method and apparatus thereof
【출원인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【대리인】	
【성명】	이영필
【대리인코드】	9-1998-000334-6
【포괄위임등록번호】	1999-009556-9
【대리인】	
【성명】	이해영
【대리인코드】	9-1999-000227-4
【포괄위임등록번호】	2000-002816-9
【발명자】	
【성명의 국문표기】	폴트닉 블라디미르
【성명의 영문표기】	PORTNYKH,Vladimir
【주소】	경기도 수원시 팔달구 영통동 황골마을 주공1단지아파트 147동 601호
【국적】	RU
【심사청구】	청구
【취지】	특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인 이영필 (인) 대리인 이해영 (인)
【수수료】	
【기본출원료】	20 면 29,000 원
【가산출원료】	28 면 28,000 원

1020020046801

출력 일자: 2003/2/6

【우선권주장료】	0 건	0 원
【심사청구료】	16 항	621,000 원
【합계】	678,000 원	
【첨부서류】	1. 요약서·명세서(도면)_1통	

1020020046801

출력 일자: 2003/2/6

【요약서】

【요약】

본 발명은 하이퍼비디오(hypervideo)의 비선형 편집방법에 관한 것으로, 구체적으로는 사용자의 편집명령을 기술하는 기술언어(descriptor language)를 제공하고 이에 대응되는 알고리즘을 제시하여 어도비 프리미어(Adobe Premier)와 같은 종래의 비선형편집기에 대응되는 태스크(task) 기반의 하이퍼비디오(hypervideo) 편집방법 및 장치에 관한 것이다. 본 발명의 편집방법은 사용자가 사용할 수 있는 가용(available) 편집명령을 초기화하는 단계, 상기 초기화된 가용 편집명령들 중에서 하나의 편집명령을 선택하는 단계, 상기 선택한 편집명령을 수행하기 위한 입력 리소스를 선택하는 단계, 상기 선택한 편집명령을 수행하고 이와 동시에 수행된 결과를 검사하는 단계 및 끝맺음 편집명령을 수행하고, 수행된 결과에 대하여 렌더링을 수행하는 단계를 구비한다. 본 발명에서 제시하는 방법을 사용하여 편집과정에서 전(previous) 단계의 수행결과를 실제 소스 파일(real source file)로서 사용할 수 있도록 하고 이를 수행하는 컴퓨터의 자원을 아낄 수 있도록 하여 편집작업에 필요한 수행시간을 단축시키고 편집 효율을 증가시키는 효과가 있다.

【대표도】

도 3

【명세서】

【발명의 명칭】

태스크 기반의 비선형 하이퍼비디오 편집방법 및 그 장치{Task oriented nonlinear hypervideo editing method and apparatus thereof}

【도면의 간단한 설명】

도 1은 타임라인 방식의 편집방법의 개념을 나타낸 도면.

도 2는 스토리보드(storyboard) 방식의 편집방법의 개념을 나타낸 도면.

도 3은 본 발명의 비디오 편집방법을 나타낸 도면.

도 4는 사용자의 편집명령을 기술하는 템플릿(template)을 나타낸 도면.

도 5는 TRANSFORM이 가지는 요소(property)와 RESOURCE를 표현하는 속성(attribute) 및 VUNION의 속성(attribute)을 나타낸 도면.

도 6은 분할(SPLIT) 편집명령 템블릿을 나타낸 도면.

도 7은 트랜지션(TRANSITION) 편집명령 템블릿을 나타낸 도면.

도 8은 병합(merge) 편집명령 템블릿을 나타낸 도면.

도 9는 삽입(insert) 편집명령 템블릿을 나타낸 도면.

도 10은 기본적인 편집 명령(editing action)을 나타낸 도표.

도 11은 결정센터(decision center)라고 불리는 화면을 나타낸 도면.

도 12는 분할(SPLIT)를 선택한 경우 나타나는 화면을 도시한 도면.

도 13은 분할(SPLIT) 동작시의 편집명령을 기술한 예를 나타낸 도면.

도 14는 삽입(INSERT) 동작시의 편집명령을 기술한 예를 나타낸 도면.

도 15는 본 발명에서 제시한 언어의 속성(attribute)을 나타낸 도면.

도 16은 BBB.AVI를 AAA.AVI 에 삽입하는 것을 표현한 도면.

도 17은 도 16의 내용을 최적화하여 표현한 도면.

도 18은 본 발명의 비선형 비디오 편집장치의 블록도면.

도 19는 편집결과를 확인(verification)하는 알고리즘을 나타내는 도면.

도 20은 본 발명에서 제시한 가상 유니온인 뷰니온(vunion)의 계층구조를 나타낸 도면.

도 21은 편집결과를 최적화(optimization)하는 알고리즘을 나타내는 도면.

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<22> 본 발명은 하이퍼비디오(hypervideo)의 비선형 편집방법에 관한 것으로, 구체적으로는 사용자의 편집명령을 기술하는 기술언어(descriptor language)를 제공하고 이에 대응되는 알고리즘을 제시하여 어도비 프리미어(Adobe Premier)와 같은 종래의 비선형편집기에 대응되는 하이퍼비디오(hypervideo) 편집방법 및 장치에 관한 것이다.

<23> 가정용 또는 전문가용(professional) 미디어 편집기로서 타임라인(timeline) 방식의 편집기 또는 스토리보드(storyboard)와 같은 형태의 편집기가 있다.

<24> 타임라인(timeline) 방식의 편집기에서는 시간에 따라서 화면을 나열한 집합(set of rows)으로 영상을 표현할 수 있으나, 다양한 비디오 클립(video

clip), 사운드 트랙(soundtrack) 그리고 효과(effect) 부분들을 차례로 쌓아놓는(stack) 형태로 리소스들이 구성되어 있으므로 시간의 흐름에 따라서 비디오를 편집하는데 있어서 불편하다. 예컨대, 긴 영화의 경우 윈도우가 매우 많이 있어야 한다는 단점이 있다.

<25> 스토리보드(storyboard) 형태의 편집기에서는 미디어 리소스(media resource)를 일련의 블록 단위로 왼쪽에서부터 오른쪽으로 위치시켜 정렬시킨다. 그리고, 클립(clip)간의 오버랩(overlap)은 T-블록(T-block)이라고 불리는 트랜지션(transition)을 넣고, 부가적인 다이얼로그 박스(dialog box)를 사용하여 조절(tunning)하여 편집을 한다.

<26> 상술한 스토리보드(storyboard) 형태의 편집기는 사용자로 하여금 스틸 이미지(still image)를 편집할 수 있도록 하고, 슬라이드 쇼(slide show)도 쉽게 편집할 수 있도록 한다. 그러나 편집하는데 있어서 시간 개념이 들어가 있지 않기 때문에 몇가지 기능이 제한되는 문제점이 있다.

<27> 한국특허공개공보 1999-55423 에서는 프로젝트 테이블(project table)을 이용하여 비디오를 처리하고 전환효과 및 시각특수효과 필터 데이터들을 관리하는 방법과 비디오의 압축상태를 풀지 않고 새 디지털 비디오(digital video)를 생성하는 방법이 개시되어 있다. 이를 위하여 프로젝트 테이블(project table)의 개념을 사용하는데, 프로젝트 테이블(project table)은 클립ID(clip ID), 스토리지(storage), 트랙 스타트(track start), 트랙 엔드(track end), 소오스 스타트(source start), 소오스 엔드(source end), 타입(type), 트랙 넘버(track number), 어플라이 클립(apply clip)이라는 요소로 구성되어 있다. 그러나 본 발명에서와 같이 사용자의 편집명령에 대한 개념이 없으며, 편집명령에 대한 구체적인 기술언어(descriptor language)를 개시하고 있지도 않다.

<28> 미국특허 5,539,527 에서도 비선형 비디오 편집 시스템이 개시되어 있다. 랜덤(random)하게 액세스할 수 있는 이미지 데이터 스토리지 유닛(image data storage unit), FIFO, 비디오 효과 유닛(video effect unit), 원하는 샷 저장유닛(desired shot storage unit)으로 구성되어 있으나 본 발명의 비선형 비디오 편집방법과는 차이가 있다.

<29> 도 1은 타임라인 방식의 편집방법의 개념을 나타낸 도면이다.

<30> 타임라인(timeline) 방식의 편집방법에서는 동영상을 시간의 흐름에 따라서 비디오 클립(video clip)들을 나열한 집합(set of rows)으로 표현하며, 다양한 비디오 클립(video clip), 사운드 트랙(soundtrack) 그리고 효과(effects) 부분들을 차례로 쌓아놓는(stack) 형태로 표현한다.

<31> 편집을 수행하기 위하여 비디오 클립(video clip)을 나타내는 아이콘(icon)을 선택하여 타임라인 인터페이스(timeline interface)에 끌고 온다(drag). 비디오 클립(video clip)은 바(bar) 형태로 표현되어 있으며, 바(bar)의 길이는 시간축 상의 실제 시간 길이를 나타낸다.

<32> 그리고, 상술한 바(bar)를 가지고 원하는 편집작업(operation)을 수행하고, 동시에 시간을 제어함으로써 다른 편집 작업도 수행할 수 있다. 마우스를 이용하여 사용자는 디스플레이를 시작할 정확한 시간을 지적할 수 있고, 부가적인 명령(additional command)을 이용하여 사용자는 영화를 몇 장면으로 분할할 수 있으며, 그 중에서 일부를 지우고 다른 것과 함께 재배치(rearrange)할 수도 있으며, 트랜지션(transition)을 정의하여 트랜지션 타임(transition time)을 쉽게 제어할 수 있다.

<33> 상술한 방법을 구현하기 위하여, 대개 일반적으로는 미리보기(previewing)를 하기 전에 렌더링(rendering)을 필요로 하는데, 시간과 밀접한 연관관계 때문에 상술한 타임라인(timeline) 방법은 스틸 이미지(still image)와 동영상 클립(video clip)을 동시에 다루는 데에는 부적합하다.

<34> 긴 비디오 클립(video clip)의 경우에는 상술한 타임라인(timeline) 방법은 부적합하다. 왜냐하면 긴 스크롤링(scrolling)을 해야 하므로 여러개의 화면(screen)이 필요하기 때문이다. 예를 들어 90 분짜리 영화의 경우 30 개 이상의 화면(screen)이 필요하다.

<35> 시간 스케일(time scale)은 변경이 가능하지만, 변경을 하게 되면 정확한 위치에서의 편집을 불가능하게 할 수 있으며, 이렇게 하기 위해서는 부가적인 윈도우가 필요하다. 타임라인(timeline) 방법은 수 개의 연속적인(consecutive) 비디오 클립(video clip)을 가지고 작업하는 것에 기초를 둔 방법이다. 즉, 만일 순서대로 있는 클립(ordered clip)을 다른 트랙에 위치시킨다면, 트랜지션(transition)을 정의하고 그 트랜지션(transition)의 지속시간(duration)을 제어하는 것도 가능하다. 다른 클립(clip)이 같은 선상에 놓여질 필요는 없으나, 그렇게 되면 트랜지션(transition)의 정의가 더 복잡하게 된다.

<36> 이와 같이 타임라인(timeline) 방법은 짧은 비디오 클립(video clip)을 편집하는 경우에 더 적절한 방법이다.

<37> 상술한 예는 타임라인(timeline) 방법과 같은 선형(linear) 편집 접근방법과 그 한계를 보여주는 예이다. 이러한 방법은 현재는 필수적인(obligatory) 것이 아니고 비선형(non-linear) 편집 시스템에 영향을 미친다. 다시 말하면, 스틸 이미지(still image)를 다루기가 어렵고, 적절한 시간 순서대로 나열하기가 어렵다는 것이다.

<38> 도 2는 스토리보드(storyboard) 방식의 편집방법의 개념을 나타낸 도면이다.

<39> 편집의 시작은 타임라인(timeline) 방법과 동일하다. 즉 사용자가 간략한 것(thumbnail)을 선택하여 스토리보드(storyboard)로 끌고 온다(drag). 스토리보드(storyboard)를 사용하는 장점은 분명함(lucidity)에 있다. 그리고 나서 사용자는 그의 미디어 리소스(media resource)를 일련의 블록으로 만들어 왼쪽에서부터 오른쪽으로 위치시켜 정렬시킨다. 클립(clip)간의 오버랩(overlap)에는 T-블록(T-block)이라고 불리는 트랜지션(transition)을 넣는데, 이는 부가적인 다이얼로그 박스(dialog box)를 사용하여 그 값을 조절(tunning)함으로써 만들 수 있다.

<40> 상술한 스토리보드(storyboard) 방식의 편집기는 사용자로 하여금 스틸 이미지(still image)를 편집할 수 있도록 하고 슬라이드 쇼(slide show)의 편집도 쉽게 할 수 있도록 한다. 그러나 편집 가능성(editing possibility)은 시간 개념이 들어가 있지 않기 때문에 제한된다.

<41> 어떤 편집명령은 효과적으로 실현될 수 있다. 예를 들어 두 개의 비디오 클립(video clip)을 병합(merge)시키고자 한다면, 그 비디오 클립(video clip)을 연속적인 셀(consecutive cell)상에 위치시키는 것이 필요하고, 그 비디오 클립(video clip)들을 선택하여 병합(merge) 명령을 실행시킨다.

<42> 그러나 이와 동시에 비디오 클립(video clip)을 2개의 독립적인(independent) 비디오 클립(video clip)으로 분할(splitting)하는 작업(operation)은 몇 가지 문제점에 직면하게 된다.

<43> 실제적으로 분할(splitting)은 스토리보드(storyboard)에서는 수행될 수 없다. 스토리보드(storyboard)에는 시간 개념(time concept)이 없기 때문이다. 일반적으로 시간과 관련된 어떠한 작업(operation)에서도 스토리보드(storyboard)를 사용한 편집방법이 타임라인(timeline)을 사용한 편집방법보다 더 복잡하게 만드는 부가적인 윈도우(additional window)를 필요로 한다.

<44> 스토리보드(storyboard) 방법은 선형 접근방법이 아니지만, 그 영향은 아직 남아 있다. 예를 들어 트랜지션(transition)은 어느 하나의 비디오 클립(video clip)에서 다른 비디오 클립(video clip)으로 단절없이 넘어가는 지시(segue)와 같이 정의된다. 그래서 오직 두 개의 비디오 클립(video clip)만이 관련되는데 이것이 선형 편집방법에서의 원칙이다. 그러나, 지금은 좋은 수행결과(performance)를 얻기 위하여 세 개의 비디오 소스(video source)를 동시에 처리하여야 한다.

【발명이 이루고자 하는 기술적 과제】

<45> 상기한 문제를 해결하기 위해 본 발명에서는 비디오를 편집하는데 있어서 사용자의 편집명령(user action)에 대한 새로운 기술언어(descriptor-language)를 제공하고 편집과정을 동적으로 제어하고 편집결과를 XTL 로 표현하는 태스트(task)에 기반을 둔 비선형 하이퍼비디오(hypervideo) 편집방법 및 장치를 제공함으로써, 편집과정(editing process)을 효율적으로 수행하도록 하는 것을 목적으로 한다.

【발명의 구성 및 작용】

<46> 상기한 목적을 이루기 위하여 본 발명에서는, 사용자가 선택하여 수행할 편집명령을 소정의 구성요소를 포함하는 확장 마크업 언어로 기술된 템플릿으로 표현하고, 상기

편집명령을 수행한 후에 렌더링을 수행하는 것을 특징으로 하는 비선형 비디오 편집방법을 제공한다.

<47> 상기한 목적을 이루기 위하여 본 발명에서는, 사용자가 사용할 수 있는 가용 편집명령을 초기화하는 단계; 상기 초기화된 가용 편집명령들 중에서 하나의 편집명령을 선택하는 단계; 상기 선택한 편집명령을 수행하기 위한 입력 리소스를 선택하는 단계; 상기 선택한 편집명령을 수행하고 이와 동시에 수행된 결과를 검사하는 단계; 및 끝맺음 편집명령을 수행하고, 수행된 결과에 대하여 렌더링을 수행하는 단계를 포함하는 비선형 비디오 편집방법을 제공한다.

<48> 상기한 목적을 이루기 위하여 본 발명에서는, 상기의 방법을 실행시키기 위한 프로그램을 기록한 컴퓨터에서 읽을 수 있는 기록매체를 제공한다.

<49> 상기한 목적을 이루기 위하여 본 발명에서는, 사용자가 사용할 수 있는 가용 편집명령을 초기화하는 편집명령 초기화부; 상기 초기화된 가용 편집명령들 중에서 사용자가 선택한 하나의 편집명령을 입력받는 편집명령 입력부; 상기 입력받은 편집명령을 수행하기 위한 입력 리소스를 입력받는 리소스 입력부; 상기 입력받은 편집명령을 수행하는 편집명령 수행부; 상기 편집명령 수행된 결과를 검사하는 결과 검사부; 및 끝맺음 편집명령을 수행하고, 상기 수행된 결과에 대하여 렌더링을 수행하는 렌더링 수행부를 포함하는 비선형 비디오 편집장치를 제공한다.

<50> 상기한 목적을 이루기 위하여 본 발명에서는, 사용자에게 가용한 편집명령을 제시하고 사용자가 선택한 편집명령을 입력받는 편집명령 입력화면을 제공하는 단계; 상기 사용자가 선택한 편집명령을 수행할 소스 파일을 제시하여 보여주는 소스파일 화면을 제공하는 단계; 및 상기 사용자가 선택한 소스파일을 가지고 상기 사용자가 선택한 편집명

령을 수행한 결과를 보여주는 결과화면 제공하는 단계를 포함하는 비선형 비디오 편집기에서의 그래픽 사용자 인터페이스 방법을 제공한다.

<51> 이하, 첨부된 도면을 참조하여 본 발명에 따른 바람직한 일실시예를 상세히 설명한다.

<52> 상술한 도 1과 도 2에서는 타임라인 방식의 편집방법과 스토리보드 방식의 편집방법을 나타내었다. 상술한 두 가지 방법에는 많은 변형(variation)이 있다. 그러나 새로운 중요한 아이디어가 부가되는 것은 아니고, 상술한 방법의 원칙을 준수한다. 그리고 현재의 소프트웨어 패키지들은 이러한 변경(modification)을 나름대로 최선의 방법으로 구현하려고 하고 있다. 이를 위해 비디오 트랜스코딩(transcoding)과 같은 기술적인 면에 주목을 하고 있으며 상술한 방법을 다소 개선하고 있다.

<53> 도 3은 본 발명의 비디오 편집방법을 나타낸 도면이다.

<54> 본 발명의 편집방법은 비디오 편집 과정을, 결정을 하는 과정(process of making decision)으로 보고, 다른 어떤 것보다 편집하는 로직(logic of editing)에 중점을 둔 방법이다.

<55> 본 발명에서는 사용자의 편집명령(user action)에 대한 새로운 기술언어(descriptor-language)를 제공한다. 이를 AVSEL(AV Station Editing Language)이라 부른다. AVSEL 은 XML(eXtended Markup Language)에 기반을 두고 있으며 따라서, 하이퍼비디오(hypervideo)라는 용어가 사용될 수 있다. 또한, 본 발명에서는 편집과정을 동적으로 제어하고 편집결과를 XTL로 표현하는 특별한 알고리즘을 제공한다. 그리고, 언어 분석기

(language analyzer)와 AVSEL 언어를 XTL로 변환시키고 그 역으로 변환시키는 문법 변환기(grammar translator)도 제공한다.

<56> 도 3과 같은 방법을 사용한 본 발명의 특징은 다음과 같다. 먼저 편집하는 중에는 렌더링(rendering)을 하지 않는다. 그리고 편집과정에서 전(previous) 단계의 수행결과는 실제 소스 파일(real source file)로서 사용할 수 있으나, 그것들은 가상적이고(virtual), 논리적인(logical) 결과이나 사용자에게는 실제 미디어 소스(media source)처럼 보인다. 따라서 컴퓨터 저장자원을 아낄 수 있도록 하고 필요한 수행시간을 단축시킴으로써 편집 효율을 크게 증가시킨다.

<57> 사용자의 편집명령은 결정 리스트(decision list)로 나눌 수 있다. 편집 시스템은 편집이 끝난 후에 어떤 종류의 렌더링 작업(rendering operation)이 수행될 것인가에 대하여 알고 있다.

<58> 렌더링 과정(rendering process)은 알고리즘적으로 수단(method)에 의해 최적화된다. 그리고, 편집결과(result of editing)는 3가지의 다른 형태로 표현될 수 있다. 그것은 수행되는 편집명령의 리스트(list of performed actions), 렌더링(rendering)을 위해 수행되는 편집명령의 논리적인 결과(logical result of performed action), 그리고 실제 미디어 파일(real media file)이다.

<59> 또한 절대적인 형태(absolute form)의 논리적인 결과와 관계된 형태의 논리적인 결과간의 변환(conversion)도 구현된다. 공개성(openness)은 새로운 비디오/오디오 변환(transform)을 위한 것이 아니라 새로운 사용자의 편집명령을 위한 것이다.

<60> 본 발명의 편집방법에서, 우선 사용자의 가용(available) 편집명령을 초기화한다(310). 사용자가 사용하는 편집명령 템플릿 언어(action template language)는 후술한다.

<61> 그리고 사용자의 상기 편집명령 템플릿 언어로 기술된 일련의 편집명령으로 기술된 편집작업(editing process)을 수행한다. 모든 사용자의 편집명령은 다음과 같은 몇 단계로 구성된다. 가용(available) 편집명령 리스트(action list)에서 편집명령을 선택한다(320). 그리고 후술하는 언어로 표현된 편집명령을 실행시키기 위한 작업(operation)을 수행한다. 이를 위하여 입력 리소스(input resource)를 선택하고(330), 편집명령을 수행하고 이와 동시에 실행된 결과를 검사한다(340).

<62> 그리고 나서 끝맺음 편집명령(finishing action)을 수행하고(350), 렌더링(rendering)을 수행한다(360).

<63> 다음은 사용자의 편집명령을 표현하기 위한 AVSEL 언어를 설명한다. 다른 카테고리(category)의 사용자는 편집의 목적이 다르고 편집과정의 다른 면에 주목한다. 이러한 편집과정은 사실상 선편집(pre-editing) 과정으로, 사용자의 요구(needs)를 인식하고(identify), 그것에 일치시켜 방법을 구상한다.

<64> 도 4는 사용자의 편집명령을 기술하는 템플릿(template)이다.

<65> 여기에서 XXX1 는 편집명령의 이름(action name)이고, N1은 입력 파라미터(input parameter)의 수, N2는 출력 파라미터(output parameter)의 수, userid는 이 연산에 대한 액세스 레벨(access level), CLSID는 이 연산에 대한 clsid, help는 편집명령에 대한 팁(small tip)을 나타낸다. 그리고, 마찬가지로 YYY1 는 입력 리소스 이름(input

resource name), P1 내지 PN은 리소스 특징의 이름(name of resource characteristics), ZZZ1 은 출력 결과 이름(output result name)을 나타낸다. 그리고, action, input, output, help, name, userid, ninput, noutput 등은 예약어(reserved words)이다.

<66> 리소스(resource)에는 미리 정의된 3개의 이름이 있는데 이는 TRANSFORM, RESOURCE, VUNION 이다. TRANSFORM은 리소스(resource)의 어떤 편집명령을 기술하는데 적용될 수 있는 트랜지션(transition)이나 효과(effects)를 의미한다.

<67> RESOURCE 는 실제 물리적으로 존재하는 영화 클립(movie clip)과 같은 리소스(resource)이다.

<68> VUNION은 실제 리소스(resource)와 트랜스폼(tranform)의 가상 유니온(union)을 의미한다.

<69> 도 5는 TRANSFORM이 가지는 요소(property)와 RESOURCE를 표현하는 속성(attribute) 및 VUNION의 속성(attribute)을 나타낸 도면이다.

<70> 다음은 몇 가지 예를 들어 상술한 템플릿(template)이 어떻게 사용되는가를 설명한다.

<71> 도 6은 분할(SPLIT) 편집명령 템블릿을 나타낸 도면이다.

<72> 분할(SPLIT) 동작은 1개의 입력 파라미터(input parameter)와 2개의 출력 파라미터(output parameter)를 갖는다. 입력 파라미터(input parameter) 및 출력 파라미터(output parameter)는 리소스(resource)에 기본을 두고 있다. 입력 파라미터(input parameter)가 실제 파일(real file)일 경우, 출력 파라미터(output parameter)는 독립적

으로 접근될 수 있는 분리된 결과(separated result)이다. 따라서 속성 fname 이 적용될 수 있다.

<73> 도 7은 트랜지션(TRANSITION) 편집명령 템블릿을 나타낸 도면이다.

<74> 도 7에서 보는 바와 같이 2개의 비디오 클립(video clip)간의 트랜지션(TRANSITION)은 3개의 입력 파라미터(input parameter)를 가지고 있다. 즉 2개의 실제 리소스(resource)와 트랜지션 표현(transition description) 그 자체이다. 출력은 유니온(union) 또는 입력 파라미터의 조합(combination)이다. 만일 작업(operation)의 결과가 전체 유니온(union)으로서 접근되어 질 수 있다면, fname이 그것을 나타내야 한다. mstart는 0 이 디폴트(default)이고, mstop 은 첫 번째 비디오 클립(video clip)과 두 번째 비디오 클립(video clip)의 길이의 합과 트랜지션(transition) 길이의 차에 의해 계산될 수 있다.

<75> 도 8은 병합(merge) 편집명령 템블릿을 나타낸 도면이다.

<76> 병합(merge) 편집명령은 두 개의 리소스(resource)를 묶어서 하나의 새로운 리소스(resource)로 만드는 것이다. 따라서 fname이 결과로 이용될 수 있다. mstart 는 0 이고. mstop 은 첫 번째 비디오 클립(video clip)과 두 번째 비디오 클립(video clip)의 길이의 합과 같다.

<77> 도 9는 삽입(insert) 편집명령 템블릿을 나타낸 도면이다.

<78> 도 9의 경우에서 두 번째 리소스(resource)인 실제 비디오 클립(video clip)이 첫 번째 비디오 클립(video clip)의 삽입 시점의 시간에 대한 상대적인 시간을 mstart 라는 속성(attribute)이 나타낸다. mstart 는 0 이 기본값(default) 이고, mstop 은 첫 번째

비디오 클립(video clip)과 두 번째 비디오 클립(video clip)의 지속시간 길이의 합이며, mlength 는 첫 번째 비디오 클립(video clip)의 mstop 과 두 번째 비디오 클립(video clip)의 mstop 의 합이다.

<79> 도 10은 기본적인 편집 명령(editing action)을 나타낸 도표이다.

<80> 이 밖에도 더 많은 사용자의 편집명령이 기술된다. 가용한(available) 편집명령 리스트(action list)는 에디터(editor)의 *.ini 파일에 저장되어 있다. 본 발명의 사용자 편집명령 템플릿(template) 언어는 아이디어 측면이나 사용분야 측면에서 XTL 과 완전히 다르다. 편집 과정(process)을 결과의 측면에서 표현하는 후자의 언어는 렌더링 과정 최적화(rendering process optimization) 접근 방법으로 표현된다. 따라서, 본 발명의 언어와 연관되어 있기는 하나 동일하지는 않다. 본 발명의 언어는 사용자의 편집명령을 기술하고 XTL 언어는 편집과정(editing process)의 결과를 표현한다.

<81> 이 단계에서 사용자의 편집명령을 초기화하는 것은 모든 사용자의 편집명령의 가용성(availability)을 다시 체크하고 주 윈도우(main window)를 설정한다.

<82> 도 11은 결정센터(decision center)라고 불리는 화면을 나타낸 도면이다.

<83> 편집 과정(editing process)은 도 11의 결정센터(decision center)라고 불리는 주 윈도우(main window)에서 시작한다. 첫 번째 단계에서 원하는 편집명령이 선택된다. 어떤 작업(operation)은 상기 화면에서 수행될 수 있다. 예를 들어 삭제(delete)는 결과를 휴지통(trash)에 넣음으로써 수행될 수 있다. 그러나 어떤 편집명령은 부가적인 단계를 필요로 한다. 예를 들어 분할(SPLIT)을 선택한 후에는 도 12와 같은 다른 윈도우가 나타난다.

<84> 도 12는 분할(SPLIT)을 선택한 경우 나타나는 화면을 도시한 도면이다.

<85> 분할(SPLIT)할 소스 또는 결과를 선택하여 주 영역으로 끌고 오거나(drag) 소스 아이콘(source icon)을 더블 클릭하여 미디어 소스(media source)를 활성화하고(activate), 분할 위치를 가리키는 스크린에서 스크롤(scroll)을 제어하고 어플라이(Apply) 버튼을 누름으로서 분할 임무(split task)가 수행된다. 분할(Split) 결과는 결과화면에 나타난다. 그러나 결과 파일을 아직까지 사용자는 얻을 수 없고 어떤 렌더링(rendering)을 수행할 것인지 시스템은 알고 있으므로 편집이 끝난후 렌더링(rendering)을 수행한다. 이러한 결과는 논리적인 결과(logical result)이지만, 사용자에게는 실제 미디어 리소스(media resource)인 것처럼 보인다.

<86> 도 13은 분할(SPLIT) 동작시의 편집명령을 기술한 예를 나타낸 도면이다.

<87> 결정 리스트(decision list)는 수행되는 편집명령의 리스트이다. 분할(SPLIT) 동작의 경우 다음(next) 정보는 결정 리스트(decision list)로 들어간다. 도 13에서 몇몇 파라미터(parameter)가 생략되어 있는데 생략된 파라미터(parameter)는 기본적인 행동(default behavior)을 하는 것으로 취급된다. 도 13의 경우 원래 비디오 클립(video clip)인 myclip.mpg 는 처음부터 시작하여 지속시간(duration) 10 까지의 첫 번째 비디오 클립(video clip)과, 10 에서 시작해서 지속시간(duration) 7 인 두 번째 비디오 클립(video clip)으로 나누어진다.

<88> 도 14는 삽입(INSERT) 동작시의 편집명령을 기술한 예를 나타낸 도면이다.

<89> vunion이 사용되기 때문에 결과는 하나의 비디오 클립(video clip)으로서 취급되지만 복잡한 구조를 갖는다. mstart는 이 유니온(union) 에서의 시간을 나타낸다. 유니온

(union)은 작은 타임라인(timeline)이며 본 발명에서 제시하는 방법은 사용자가 시간의 개념을 다룰 수 있도록 한다. mstart 의 계산 알고리즘은 편집명령에 따라서 달라지며, 편집명령 그 자체뿐만 아니라, 그 편집명령을 실행시키는 프로세스(process)에 의해서도 제공되어야 한다. 그러나 다음과 같이 mstart를 계산하는 일반적인 알고리즘도 있다.

<90> (1) 유니온(union)에 있는 첫 번째 비디오 클립(video clip)에서 mstart 는 0 이 됨.

<91> (2) 두 번째 비디오 클립(video clip)에서 mstart는 첫 번째 비디오 클립(video clip)의 지속시간(duration)과 동일함.

<92> (3) 모든 다음 비디오 클립(video clip)에 대해서 mstart 는 바로 전 비디오 클립(video clip)의 지속시간(duration)에 의하여 증가됨.

<93> 끝맺음(finishing) 편집명령의 확인(confirmation) 후에 사용자는 다른 편집명령이나 동일한 편집명령을 수행할 수 있다. 모든 편집명령은 편집명령 리스트(action list)에 부가된다. 본 발명의 방법을 따르면 수행결과가 시간지연 없이 즉시 이용할 수 있으나, 그 결과는 가상적인 것이고 사용자의 편집명령과 그 결과는 본 발명에서 제시하는 언어를 사용하여 표현되고 특별한 명령(special command)에 의하여 실제적인 것으로 나타난다(export).

<94> 사용자는 결정 리스트(decision list)를 직접 생성할 수 있다. 왜냐하면 결정 리스트(decision list)는 오픈 소스(open source)이며 텍스트 에디터(text editor)에 의하여 다루어질 수 있기 때문이다. 또한 언어 표현(language description)도 이용할 수 있으

며, 다른 도구(tool)도 결정 리스트(decision list)를 자동적으로 생성하는데 사용할 수 있다.

<95> 만일 사용자가 실제적인 결과를 얻고자 한다면 모든 동작(operation)을 수행한 후에 단지 Export 버튼을 누르면 된다. 이렇게 되면 실제적인 소스(real source)가 만들어지지만 컴퓨터의 자원과 별도의 시간을 소비한다. 그리고, 디폴트(default)로 편집을 끝낸 후에 사용자 편집명령 리스트(action list)를 얻을 수 있다.

<96> 끝맺음 편집명령(finishing action)을 수행한 후에 렌더링(rendering)을 수행한다. 사용자는 비록 사용자에게는 최적의 방법이고 렌더링(rendering)하는 프로세스에는 최적이 아니라 하더라도, 어떤 결과를 최적의 방법이 아닌 다른 방법으로 얻을 수 있다. 그래서, 얻어지는 결과를 다른 방법으로 표현할 필요가 있다. 다음과 같이 3가지의 중요한 다른 포맷을 사용할 수 있다. 즉 사용자 편집명령의 전사(transcription)를 뜻하는 사용자의 결정 리스트(decision list), 결과 요약(result summary) 및 실제 미디어 파일이다. 사용자의 결정 리스트(decision list)는 상술한 바와 같고 실제 미디어 파일은 종래의 미디어 파일과 동일하다.

<97> 결과 요약(result summary)은 XML에 기반을 둔 포맷을 사용하며 소스를 링크(link)로 다룬다. 사용자의 결정 리스트(decision list)와 다른 점은 결과가 어떻게 얻어졌는가에 대한 정보는 무시하고, 결과 자체에만 중점을 둔다는 것이다. 이러한 포맷에서 사용자의 모든 편집명령은 요약될 수 있고, 모든 가상의 결과가 어떻게 얻어졌는가에 대한 기술에 의하여 대체될 수 있다. 내용(context)은 시간상으로 mstart 값에 의하여 정렬(sorted)되며 여분의 네스팅(nesting)은 제거된다. 이러한 포맷은 렌더링(rendering)하는 데에도 준비되며 XTL로 변환 가능하다. 이 변환된 파일에 의하여 사용자의 편집명령

은 복구된다. 편집결과는 사용자의 편집명령의 수행결과이며 따라서 본 발명에서 제시하는 언어는 이러한 TRANSFORM, RESOURCE 및 VUNION을 다룬다. 각각의 속성(attribute)은 다음과 같다.

<98> TRANSFORM = {CLSID, mute, mstart, mstop}

<99> RESOURCE = {start, stop, mstart, mstop, src, fname, stream, mute}

<100> VUNION = {mstart, mstop, mute, fname}

<101> 상술한 것들은 사용자 편집명령을 기술하는 데에 있어서 기본적인 구성요소(component)이다. 트랜스폼(transform)의 값(property)이 변화할 수 있기 때문에 오직 중요한 구조는 고정된다.

<102> 본 발명에서 제시하는 언어의 기본적인 시맨틱(semantic)은 XTL의 시맨틱(semantic)과 동일하다. 따라서 본 발명에서 구체적으로 설명하지는 않는다. 마이크로소프트사의 XTL 언어와 비교하여 분명한 장점이 있다. 그 장점들 중에서 어떤 것은 기술적인 것이나, 어떤 것들은 완전히 새로운 것이다. 이러한 기술적인 아이디어는 본 발명에서 제시하는 언어의 속성(attribute)에 포함되어 있다.

<103> 도 15는 본 발명에서 제시한 언어의 속성(attribute)을 나타낸 도면이다.

<104> " 가 필요하지 않고 ' 를 사용하거나 부호를 사용하지 않아도 무방하다. RESOURCE는 XTL 에서의 CLIP 과 유사하나 몇 가지 불필요한 속성(attribute)을 CLIP 에서 제거한 것이다. TRANSFORM 은 XTL 에서 편집 프로세스를 간단하게 하기 위하여 TRANSITION과 EFFECTS를 조합(combination)하였다. VUNION 은 새로운 속성(attribute)이다. VUNION은

mstop 과 mstart 속성(attribute)을 가지고 있다. 그것들은 현재의 새로운 기술이며 렌더링(rendering) 프로세스의 최적화를 수행하는데 있어서는 중대한(critical) 것이다.

<105> 간단한 시나리오를 예를 들어 본다. AAA.AVI와 BBB.AVI 라는 파일이 있고 각각 지속시간(duration)이 10과 23이며, BBB.AVI를 AAA.AVI에 4에서 시작하여 삽입(insert)하고, AAA.AVI의 시작에서 첫 번째 1초까지를 자르고 렌더링(rendering)하고, AAA.AVI의 마지막 2초를 잘라서 렌더링(rendering)하고자 하는 경우에 도 16과 같이 표현된다.

<106> 도 16은 상술한 BBB.AVI를 AAA.AVI 에 삽입하는 것을 표현한 도면이다.

<107> 모든 네스티드(nested) VUNION은 페어런트(parent) 보다 앞서 렌더링(rendering)되어야 한다. 왜냐하면 리소스(resource)를 제공하고 유니온(union)의 계층(hierarchy)이 렌더링 프로세스(rendering process)를 최적화하기 때문이다.

<108> 도 17은 도 16의 내용을 최적화한 표현이다.

<109> 최적화(optimization)의 효과는, 최적화가 실시간으로 수행된다면 최적화에 필요한 시간의 형태로 표현된다. 최적화하지 않은 결과는 (31-1)+(33-0)=63 이 되며, 최적화한 결과는 (31-1)=30 이 된다. 따라서 거의 두 배의 효과가 있으므로 최적화(optimization)는 효과가 있다.

<110> 도 18은 본 발명의 비선형 비디오 편집장치의 블록도면이다.

<111> 본 발명의 비디오 편집장치는 편집명령 초기화부(1810), 편집명령 입력부(1820), 리소스 입력부(1830), 편집명령 수행부(1840), 결과 검사부(1850), 렌더링 수행부(1860)를 구비하고 있다.

<112> 편집명령 초기화부(1810)는 사용자가 사용할 수 있는 가용 편집명령을 초기화한다. 편집명령 입력부(1820)는 상기 초기화된 가용 편집명령들 중에서 사용자가 선택한 하나의 편집명령을 입력받는다. 리소스 입력부(1830)는 상기 입력받은 편집명령을 수행하기 위한 입력 리소스를 입력받는다. 편집명령 수행부(1840)는 상기 입력받은 편집명령을 수행한다. 결과 검사부(1850)는 상기 입력받은 편집명령을 수행하고 이와 동시에 수행된 결과를 검사한다. 그리고, 렌더링 수행부(1860)는 끝맺음 편집명령을 수행하고, 상기 수행된 결과에 대하여 렌더링을 수행한다.

<113> 본 편집장치에서 상기 편집명령은 상기 수행될 편집명령의 이름정보, 상기 편집명령에서 사용되는 입력 파라미터의 수 정보, 상기 편집명령의 결과로서 출력되는 출력 파라미터의 수 정보, 상기 수행될 편집명령에 대한 액세스 레벨 정보를 포함한다.

<114> 도 19는 편집결과를 확인(verification)하는 알고리즘을 나타내는 도면이다.

<115> 모든 프로그램 언어는 특별한 의미를 가지고 문법적인 순서를 가지는 집합들을 가지고 있다. 최근에 매우 유용하게 사용되고 있는 마크 업 언어(markup language)의 하나인 XML도 그러한 특성을 가지고 있다. 그리고 컴파일러(complier)는 우선, 입력 정보에 대한 의미의 일관성을 체크하고 있다. 그러한 특성을 본 발명에서 제안한 새로운 언어도 가지고 있을 뿐만 아니라, 다음과 같은 고유의 특성을 가지고 있다.

<116> 우선 사용자의 편집명령에 대해 임의의 접근(random access)이 가능하다. 그리고 사용자의 편집명령은 결과에 의해서 완성된다. 도 19에서 설명한 확인(verification) 알고리즘은 상기 사용자의 편집명령을 검증하기 위한 알고리즘을 제공하고 있다.

<117> 도 19를 설명하기 위하여 다음과 같이 몇가지 용어를 정의한다. C 는 리소스 파일을 의미하는 것으로 $C = \{c_j\}_j$ 이다. 사용자의 편집명령은 하나의 함수로써 다음 수학식 1과 같이 정의된다.

<118> 【수학식 1】 $A_{11}, A_{12}, \ldots, A_{ij}, \ldots, A_{nm} : C^n \rightarrow C^m$

<119> 여기서 $C^n = \{(c^1, c^2, \ldots, c^j, \ldots, c^n)\}$, $c^j \in C$, $j = 1..n$, $C^m = \{(c^1, c^2, \ldots, c^i, \ldots, c^m)\}$, $c^i \in C$, $i = 1..m$ 이다.

<120> 그리고 $Z = \{A_{11}, A_{11}, A_{11}, A_{11}\}$를 편집명령의 집합으로 정의하자. 또한 A_K^{-1} 는 역함수이고 다음 수학식 2와 같이 정의된다.

<121> 【수학식 2】 $A_K^{-1} \in (\cup A_i) \cup C$, $I= 1...k-1$

<122> 확인(verification) 과정은 다음과 같다. 우선, 초기화를 수행한다(1910). 초기화 단계에서는 Z={}로 하고 사용자 편집명령의 수를 나타내는 N을 입력받고 K를 1로 설정한다. 그리고, K<=N인지 판단한다(1920). 판단결과 K<=N 이면 A_K^{-1}을 계산하고(1930), 계산한 결과가 실제 파일(real file)인지 검사한다(1940).

<123> 만일 실제 파일(real file)이면 리소스(resource)의 가용성(availability)을 체크하고 리소스 파일 C를 처리하며(1950), 그렇지 않으면 이 파일이 Z의 엘리먼트(element)인가를 검사한다(1960). 그리고 난 후 리소스(resource)가 가용한지 체크하여(1970), 가용하면 다시 다음 K에 대하여 K<=N을 비교하고 가용하지 않으면 경고 메시지를 출력한다(1980). 이와 같은 단계를 K가 N이 될 때까지 수행하여 K가 N이 되면 Z를 계산한다(1990). Z는 Z와 A_K의 교집합(union)으로 구해진다.

<124> 도 20은 본 발명에서 제시한 가상 유니온인 뷰니온(vunion)의 계층구조를 나타낸 도면이다.

<125> 도면에서 보는 바와 같이 가상 유니온은 계층적으로 구성되어 있으며 그 속성값(property)으로 mstart, mstop 등을 가진다.

<126> 도 21은 편집결과를 최적화(optimization)하는 알고리즘을 나타내는 도면이다.

<127> 렌더링 단계는 아주 많은 시간을 요구한다는 측면에서 편집과정에 있어서 가장 중요한 단계이다. 그래서 이러한 렌더링 단계를 최적화하는 것은 매우 중요한 과정이다. 이러한 렌더링 시간을 줄이는 방법은 몇가지 알려져 있다. 그중 잘 알려진 방법중의 하나는 압축(encoding)하는 기술을 향상시키는 방법이다. 그러나 이러한 방법은 시간의 단축이 크게 일어나지 않는다. 하지만 본 발명의 최적화(optimization) 알고리즘은 렌더링 시간을 단축시키는데 효과가 뛰어나다.

<128> 최적화는 다음의 과정에 의해서 수행된다.

<129> AVSEL 스트링을 null 로 한 다음, 첫 번째 엘리먼트(element)를 입력받는다(2110), 상기 엘리먼트(element)가 표준 엘리먼트(standard element)인가를 판단해서(2120) 표준 엘리먼트(standard element)이면 AVSEL 스트링을 다시 입력받는다(2130).

<130> 만일 표준 엘리먼트(standard element)가 아니면 mstart를 계산한다(2140).

<131> mstart의 계산은 다음과 같은 방법에 의한다.

<132> Klopt = start로 하고 첫 번째 nested element로 이동한다. 그리고, 다음의 while 문을 수행한다.

<133> while(Klopt > mstop - mstart)

<134> {

<135> Klopt = Klopt - (mstop - mstart)

<136> 현재의 엘리먼트 삭제(Delete current element)

<137> 처음 nested element로 이동

<138> }

<139> 상기 while 루프를 빠져나오면, 모든 nested element에 대해서 parent의 mstart와 mstop을 엘리먼트(element)의 mstart와 mstop으로 조정한다. 그리고 나서, mstop을 계산한다(2150). mstop의 계산은 다음의 과정에 의한다.

<140> K2opt = mstop 으로 하고, 마지막 nested element 로 이동한 후에 다음의 while 문을 수행한다.

<141> while(K2opt < mstart)

<142> {

<143> 현재의 엘리먼트 삭제(Delete current element)

<144> 마지막 nested element로 이동

<145> }

<146> 그리고 나서 상기 K2opt를 mstop에 저장한다.

<147> mstart와 mstop에 대한 계산이 끝난 후에 자식 엘리먼트(child element)를 부모 엘리먼트(parent element)로 만들고(2160), 상기 AVSEL 스트링을 출력한다(2170).

<148> 그리고, 이러한 알고리듬을 모든 엘리먼트들에 대해서 반복 실행함으로써, AVSEL 스트링을 얻어낸다. 즉, 다음 엘리먼트를 입력받아, 상술한 과정을 반복해서 실행한다.

<149> 한편, 상술한 본 발명의 실시예들은 컴퓨터에서 실행될 수 있는 프로그램으로 작성 가능하고, 컴퓨터로 읽을 수 있는 기록매체를 이용하여 상기 프로그램을 동작시키는 범용 디지털 컴퓨터에서 구현될 수 있다.

<150> 상기 컴퓨터로 읽을 수 있는 기록매체는 마그네틱 저장매체(예를 들면, 롬, 플로피 디스크, 하드디스크 등), 광학적 판독 매체(예를 들면, 씨디롬, 디브이디 등) 및 캐리어 웨이브(예를 들면, 인터넷을 통한 전송)와 같은 저장매체를 포함한다.

<151> 이제까지 본 발명에 대하여 그 바람직한 실시예들을 중심으로 살펴보았다. 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

【발명의 효과】

<152> 상술한 바와 같이 본 발명은, 사용자의 편집명령(user action)에 대한 새로운 기술언어(descriptor-language)를 제공하고 편집과정을 동적으로 제어하고 편집결과를 XTL로 표현하는 특별한 알고리즘을 제공함으로써, 편집과정에서 전(previous) 단계의 수행결과를 실제 소스 파일(real source file)로서 사용할 수 있도록 하고 컴퓨터 자원을 아낄 수 있도록 하여 필요한 수행시간을 단축시키고 편집 효율을 증가시키는 효과가 있다.

<153> 본 발명에서 제공하는 비디오 편집방법에서는, 사용자는 우선 문제를 확인하고(identify), 입력되는 정보를 분명히 이해하여 결과를 얻기 위한 특별한 과정(procedure)에 따라서 태스크(task)를 수행한다. 그리고, 본 발명에서는 트랜지션(transition)과 이펙트(effect)를 가진 결과를 렌더링(rendering)없이 프리뷰(preview)를 할 수 있다. 아도브 프리미어(Adobe Premier)에서는 자동적으로 이러한 프리뷰(preview)를 보여줄 수 없다.

<154> 본 발명은 공개된 구조(open architecture)로 구성되어 있다. 사용자는 결정 리스트(decision list)를 직접 액세스하고, 어떤 리소스(resource)도 추가하거나 제거(add/remove)할 수 있다. 그리고, 사용자는 어떤 포맷으로 개시(description)할 수 있다면, 사용자가 미리 정의한 변환(transformation)을 사용할 수도 있다.

<155> 그리고, 본 발명은 원하는 결과를 얻기 위하여 사용자가 같은 편집명령을 여러 번 반복할 수 있도록 허용하고, 심지어는 이러한 단계들을 반복하지 않고 전에 수행되어진 몇 단계의 어떤 작업(operation)을 반복할 수 있도록 허용한다.

<156> 편집하고(editing) 프리뷰잉(previewing) 단계(stage)는 결합되어 있다. 그러나 렌더링(rendering)은 분리되어 있으며 반드시 필요한 것은 아니며, 편집(editing)의 마지막 단계(stage)에서 수행될 수 있다. 중간(intervening) 결과는 링크(link)로 저장되어 있는데, 이는 하이퍼비디오(hipervideo) 편집을 의미하며, 인코딩 프로시저(procedure)를 최적화하는데(optimize) 도움을 준다. 그리고, 동시에 다른 포맷의 리소스들은 하드디스크, 메모리와 같은 컴퓨터 리소스(resource)들을 줄여 편집과정(editing process)을 효율적으로 수행하도록 한다.

【특허청구범위】

【청구항 1】

(a) 사용자가 선택하여 수행할 편집명령을 소정의 구성요소를 포함하는 확장 마크업 언어로 기술된 템플릿으로 표현하는 단계; 및

(b) 상기 편집명령을 수행한 후에 렌더링을 수행하는 단계를 포함하는 것을 특징으로 하는 비선형 비디오 편집방법.

【청구항 2】

제1항에 있어서, 소정의 구성요소는

상기 수행될 편집명령의 이름 정보, 상기 편집명령에서 사용되는 입력 파라미터의 수 정보, 상기 편집명령 수행의 결과로서 출력되는 출력 파라미터의 수 정보, 상기 수행될 편집명령에 대한 액세스 레벨 정보를 포함하는 것을 특징으로 하는 비선형 비디오 편집방법.

【청구항 3】

(a) 사용자가 사용할 수 있는 가용 편집명령을 초기화하는 단계;

(b) 상기 초기화된 가용 편집명령들 중에서 하나의 편집명령을 선택하는 단계;

(c) 상기 선택한 편집명령을 수행하기 위한 입력 리소스를 선택하는 단계;

(d) 상기 선택한 편집명령을 수행하고 이와 동시에 수행된 결과를 검사하는 단계; 및

(e) 끝맺음 편집명령을 수행하고, 수행된 결과에 대하여 렌더링을 수행하는 단계를 포함하는 비선형 비디오 편집방법.

【청구항 4】

제3항에 있어서, 상기 편집명령은

상기 수행될 편집명령의 이름 정보, 상기 편집명령에서 사용되는 입력 파라미터의 수 정보, 상기 편집명령의 결과로서 출력되는 출력 파라미터의 수 정보, 상기 수행될 편집명령에 대한 액세스 레벨 정보를 포함하여 표현하는 것을 특징으로 하는 비선형 비디오 편집방법.

【청구항 5】

제3항에 있어서, 상기 리소스는

상기 리소스에 대해서 수행되는 편집명령을 기술하는데 적용되는 트랜지션과 효과를 의미하는 트랜스폼, 실제 물리적으로 존재하는 비디오 클립, 상기 트랜스폼과 실제 비디오 클립의 가상 유니온을 포함하는 것을 특징으로 하는 비선형 비디오 편집방법.

【청구항 6】

제3항에 있어서, 상기 (e) 단계에서 렌더링을 수행하는 것은

그 결과가 상기 수행된 편집명령의 리스트, 상기 렌더링을 수행하기 전에 수행된 편집명령의 논리적인 결과 및 실제 비디오 파일인 것을 특징으로 하는 비선형 비디오 편집방법.

【청구항 7】

제3항에 있어서, 상기 편집명령은

임포트 리소스 기능, 클로우즈 리소스 기능, 편집 결과의 삭제 기능, 편집 결과의 익스포트 기능, 클립의 분할 기능, 클립의 병합 기능, 클립의 삽입 기능을 포함하는 것을 특징으로 하는 비선형 비디오 편집방법.

【청구항 8】

제7항에 있어서, 상기 임포트 리소스 기능은

외부의 로컬 파일이나 디지털 카메라 또는 인터넷 스트리밍 소스로부터 데이터를 입력받아 편집하는 것을 특징으로 하는 비선형 비디오 편집방법.

【청구항 9】

제7항에 있어서, 상기 편집 결과의 익스포트 기능은

상기 편집결과를 MPEG-2와 같이 사용자가 지정한 특별한 형태의 데이터 포맷으로 저장하는 것을 특징으로 하는 비선형 비디오 편집방법.

【청구항 10】

제1항 내지 제9항 중 어느 한 항에 기재된 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

【청구항 11】

사용자가 사용할 수 있는 가용 편집명령을 초기화하는 편집명령 초기화부;

상기 초기화된 가용 편집명령들 중에서 사용자가 선택한 하나의 편집명령을 입력받는 편집명령 입력부;

상기 입력받은 편집명령을 수행하기 위한 입력 리소스를 입력받는 리소스 입력부;

상기 입력받은 편집명령을 수행하는 편집명령 수행부;

상기 편집명령 수행된 결과를 검사하는 결과 검사부; 및

끝맺음 편집명령을 수행하고, 상기 수행된 결과에 대하여 렌더링을 수행하는 렌더링 수행부를 포함하는 비선형 비디오 편집장치.

【청구항 12】

제11항에 있어서, 상기 편집명령은

상기 수행될 편집명령의 이름정보, 상기 편집명령에서 사용되는 입력 파라미터의 수 정보, 상기 편집명령의 결과로서 출력되는 출력 파라미터의 수 정보, 상기 수행될 편집명령에 대한 액세스 레벨 정보를 포함하여 표현하는 것을 특징으로 하는 비선형 비디오 편집장치.

【청구항 13】

(a) 사용자에게 가용한 편집명령을 제시하고 사용자가 선택한 편집명령을 입력받는 편집명령 입력화면을 제공하는 단계;

(b) 상기 사용자가 선택한 편집명령을 수행할 소스 파일을 제시하여 보여주는 소스파일 화면을 제공하는 단계; 및

(c) 상기 사용자가 선택한 소스파일을 가지고 상기 사용자가 선택한 편집명령을 수행한 결과를 보여주는 결과화면 제공하는 단계를 포함하는 비선형 비디오 편집기에서의 그래픽 사용자 인터페이스 방법.

【청구항 14】

제13항에 있어서, 상기 (a) 단계에서의 편집명령은

상기 편집명령을 수행할 입력 파일을 나타내는 리소스, 상기 리소스에 대해서 편집명령을 기술하는데 사용되는 트랜지션이나 효과를 나타내는 트랜스폼, 상기 리소스와 상기 트랜스폼의 가상 유니온을 나타내는 뷰니온으로 표현하는 것을 특징으로 하는 비선형 비디오 편집기에서의 그래픽 사용자 인터페이스 방법.

【청구항 15】

제14항에 있어서, 상기 리소스는

상기 리소스에 대한 디스플레이 시작시간 정보, 상기 리소스에 대한 디스플레이 정지시간 정보, 상기 리소스를 편집하는 시작시간 정보, 상기 리소스를 편집하는 정지시간 정보, 상기 리소스 파일 이름정보, 사운드의 가용여부를 나타내는 정보를 포함하여 표현하는 것을 특징으로 하는 비선형 비디오 편집기에서의 그래픽 사용자 인터페이스 방법.

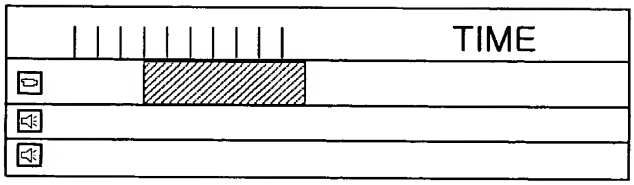
【청구항 16】

제14항에 있어서, 상기 뷰니온은

상기 리소스를 편집하는 시작시간 정보, 상기 리소스를 편집하는 정지시간 정보, 사운드의 가용여부를 나타내는 정보를 포함하여 표현하는 것을 특징으로 하는 비선형 비디오 편집기에서의 그래픽 사용자 인터페이스 방법.

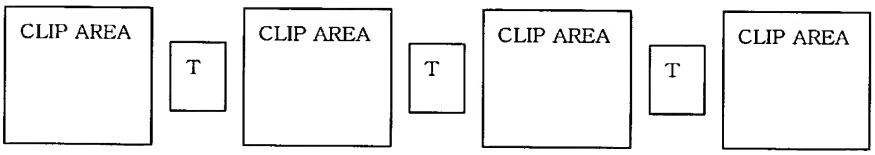
【도면】

【도 1】

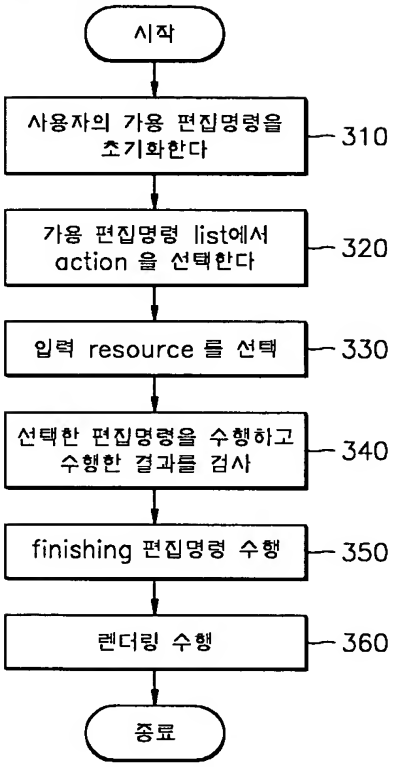


TIME	1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....
Video track 1	############
Transition	$$$$$
Video track 2	################
Transition	
Video track 3	
....	
Audio track 1	**********
Audio track 2	
Additional tracks	

【도 2】



【도 3】


시작
사용자의 가용 편집명령을 초기화한다
310
가용 편집명령 list에서 action 을 선택한다
320
입력 resource 를 선택
330
선택한 편집명령을 수행하고 수행한 결과를 검사
340
finishing 편집명령 수행
350
렌더링 수행
360
종료

【도 4】

```
<action name="XXX1"  ninput="N1"  noutput="N2"  userid="XXX2" CLSID="clsid
 help="XXX3">
  <input name="YYY1">
       <name="P1"/>
       <name="P2"/>
          .....
       <name="PN"/>
  </input>
  <input name="YYY2">
       <name="P1"/>
       <name="P2"/>
          .....
       <name="PN"/>
  </input>

  ......

  <output name="ZZZ1">
       <name="P1"/>
       <name="P2"/>
          .....
       <name="PN"/>
  </output>
  <output name="YYY1">
       <name="P1"/>
       <name="P2"/>
          .....
       <name="PN"/>
  </output>

  ......

</action>
```

【도 5】

템플릿	속성	설명
TRANSFORM 템플릿	CLSID	전역 아이디 수(unique global identification number)
	mute	트랜지션(transition)동안 사운드가 available 한지 안 한지의 여부 (TRUE or FALSE 값이 됨)
	mstart	main stream 에 대하여 transform 의 시작시간 (start time of transform relative to main stream)
	mstop	main stream 에 대하여 transform 의 정지시간 (stop time of transform relative to main stream)
RESOURCE 템플릿	start	실제 클립의 시작에 대하여 디스플레이 시작시간 (start time for displaying relative to the beginning of real clip)
	stop	실제 클립에 대하여 디스플레이 정지시간 (stop time for displaying relative to real clip)
	mstart	편집모델(editing model)에 대하여 클립을 디스플레이하는 시작시간 (start time of displaying clip relative to editing model)
	mstop	편집모델(editing model)에 대하여 클립을 디스플레이하는 정지시간 (stop time of displaying clip relative to editing model)
	src	리소스 파일이름(resource file name)
	fname	목적 파일이름이 원래파일 이름과 다를 경우, 적용가능한 파일이름 (file name applicable if destination file name differs from origin one)
	stream	원래 파일에서의 스트림 수(number of stream in original file)
	mute	사운드가 available 한가에 따라 TRUE 또는 FALSE 값을 가짐 (TRUE or FALSE depends if sound is available or not)
VUNION 템플릿	mstart	편집모델(editing model)에 대하여 클립을 디스플레이하는 시작시간 (start time of displaying clip relative to editing model)
	mstop	편집모델(editing model)에 대하여 클립을 디스플레이하는 정지시간 (stop time of displaying clip relative to editing model)
	mute	사운드가 available 한가에 따라 TRUE 또는 FALSE 값을 가짐 (TRUE or FALSE depends if sound is available or not)
	fname	목적파일 이름(destination file if applicable)

【도 6】

```
<action name="SPLIT" ninput="1" noutput="2" userid="0"  CLSID="ID_SPLIT"
 help="some tips">
  <input name="RESOURCE">
       <name="framerate"/>
       <name="start"/>
       <name="stop"/>
       <name="src"/>
       <name="stream"/>
  </input>

  <output name="VUNION">
       <name="framerate"/>
       <name="start"/>
       <name="stop"/>
       <name="stream"/>
       <name="src"/>
       <name="fname"/>
  </output>
  <output name="VUNION">
       <name="start"/>
       <name="framerate"/>
       <name="start"/>
       <name="stop"/>
       <name="src"/>
       <name="stream"/>
       <name="fname"/>
  </output>
</action>
```

【도 7】

```
<action name="TRANSITION" ninput="3" noutput="1" userid="0"
 CLSID="ID_TRANSITION" help="some tips">
  <input name="clip">
       <name="framerate"/>
       <name="start"/>
       <name="stop"/>
       <name="src"/>
       <name="stream"/>
  </input>
  <input name="clip">
       <name="framerate"/>
       <name="start"/>
       <name="stop"/>
       <name="mstart"/>
       <name="src"/>
       <name="stream"/>
  </input>
  <input name="transition">
       <name="clsid"/>
       <name="mute"/>
       <name="start"/>
       <name="stop"/>
  </input>
  <output name="vunion">
       <name="lock"/>
       <name="mute"/>
       <name="mstart"/>
       <name="mstop"/>
       <name="fname"/>
  </output>
</action>
```

【도 8】

```
<action name="MERGE" ninput="2" noutput="1" userid="0" CLSID="ID_MERGE"
 help="some tips">
  <input name="clip">
       <name="framerate"/>
       <name="start"/>
       <name="stop"/>
       <name="src"/>
       <name="stream"/>
  </input>
  <input name="clip">
       <name="framerate"/>
       <name="start"/>
       <name="stop"/>
       <name="src"/>
       <name="stream"/>
  </input>
  <output name="vunion">
       <name="lock"/>
       <name="mute"/>
       <name="mstart"/>
       <name="mstop"/>
       <name="fname"/>
  </output>
</action>
```

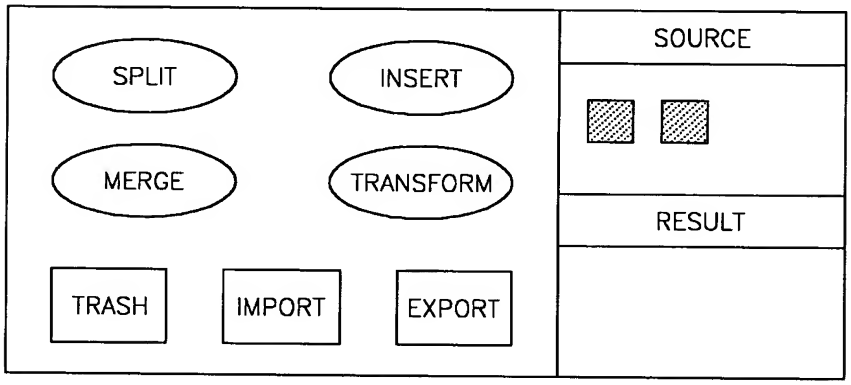
【도 9】

```
<action name="INSERT" ninput="2" noutput="1" userid="0"  CLSID="ID_INSERT"
 help="some tips">
  <input name="clip">
       <name="framerate"/>
       <name="start"/>
       <name="stop"/>
       <name="src"/>
       <name="stream"/>
  </input>
  <input name="clip">
       <name="framerate"/>
       <name="start"/>
       <name="mstart"/>
       <name="stop"/>
       <name="src"/>
       <name="stream"/>
  </input>
  <output name="vunion">
       <name="lock"/>
       <name="mute"/>
       <name="mstart"/>
       <name="mstop"/>
       <name="fname"/>
  </output>
</action>
```

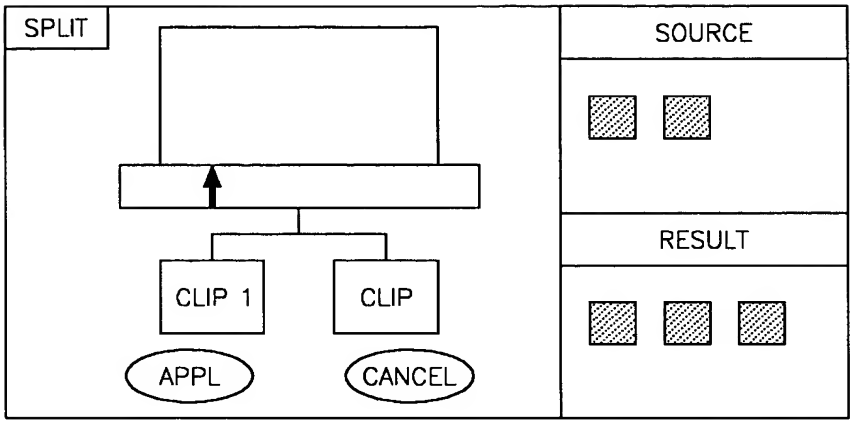
【도 10】

번호	편집명령 이름 (action name)	설명
1	Import resource	편집 프로세스(editing process)는 미디어 리소스를 다루는데, 이는 로컬 파일(local file), DV, 카메라 카세트, 인터넷 스트리밍 리소스 등이 있다. Import 리소스는 편집하는데 이용할 수 있다는 (availability) 마크(mark)를 하고, 그 리소스에 대한 특별한 리소스 별명(moniker)과 정보를 받는 것을 의미한다.
2	Close resource	리소스 별명(moniker)을 free 시키고, 계속되는 단계에서 직접 편집하는데 리소스를 이용할 수 없도록(unavailable)한다.
3	Delete editing result	중요하지 않은 결과는 지운다. 따라서 사용자가 오직 중요한 결과만 저장하도록 도와준다.
4	Export editing result	편집 행동(action)을 수행한 후 결과는 특별한 형태로 MPEG 2와 같은 특별한 포맷으로 저장한다.
5	Split clip	전체 클립을 어떤 시간에서부터 어떤 시간까지 끝나는 몇 개의 조각으로 나눈다.
6	Merge clips	몇 개의 다른 클립을 하나의 클립으로 병합한다.
7	Insert clip	소스(source)라고 불리는 하나의 클립을 어떤 시간에 타겟(target)이라고 부르는 다른 클립에 삽입한다.
8	Define transform	사용자가 효과(effect)나 트랜지션(transition)과 같은 트랜스폼(transform) 을 정의하도록 한다.

【도 11】



【도 12】



【도 13】

```
<action name="SPLIT" ninput="1" noutput="2" userid="0" CLSID="ID_SPLIT"
 help="some tips">
  <input name="clip">
       <start="0"/>
       <stop="17"/>
       <src="myclip.mpg"/>
  </input>
  <output name="clip">
       <start="0"/>
       <stop="10"/>
       <src="myclip.mpg"/>
  </output>
  <output name="clip">
       <start="10"/>
       <stop="17"/>
       <src="myclip.mpg"/>
  </output>
</action>
```

【도 14】

```
<action name="INSERT" ninput="2" noutput="1" userid="0" CLSID="ID_INSERT"
 help="some tips">
  <input name="clip">
       <start="0"/>
       <stop="10"/>
       <src = "targetclip.mpg"/>
  </input>
  <input name="clip">
       <start="0"/>
       <stop="27"/>
       <src = "clip.mpg"/>
  </input>

<output name="vunion">
  <output name="clip">
       <start="0"/>
       <stop="4"/>
       <mstrat="0"/>
       <src = "targetclip"/>
  </output>
  <output name="clip">
       <mstart="4"/>
       <start="0"/>
       <stop="27"/>
       <src = "clip.mpg"/>
  </output>
  <output name="clip">
       <start="4"/>
       <stop="10"/>
       <mstart="31"/>
       <src = "targetclip.mpg"/>
  </output>
</output>
</action>
```

【도 15】

<RESOURCE> <start ="..." /> <stop = "..."/> <mstart = "..."/> <mstop = "..."/> <src = "..."/> <fname = "..."/> <stream= "..."/> <mute="..."/> </RESOURCE>	<TRANSFORM> <clsid="..."/> <mute="..."/> <mstart="..."/> <mstop="..."/> ..... </TRANSFORM>	<VUNION> <mstart="..."/> <mstop="..."/> <mute="..."/> <fname="..."/> </VUNION>

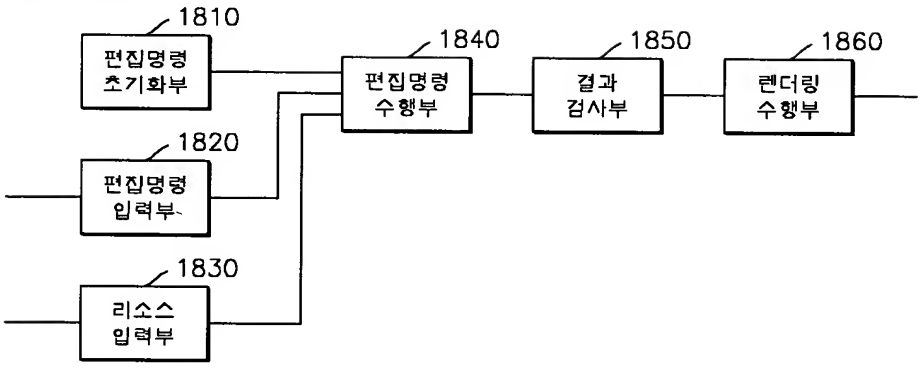
【도 16】

```
<VUNION>
    <mstart=1/>
    <mstop=31/>
    <VUNION/>
        <mstart=0/>
        <mstop=33/>
        <RESOURCE>
                <src="AAA.AVI"/>
                <mstart=0/>
                <mstop=4/>
        </RESOURCE>
        <RESOURCE>
                <src="BBB.AVI"/>
                <mstart=0/>
                <mstop=23/>
                <start=4/>
        </RESOURCE>
        <RESOURCE>
                <src="AAA.AVI"/>
                <mstart=27/>
                <mstop=33/>
        </RESOURCE>
    </VUNION>
</VUNION>
```

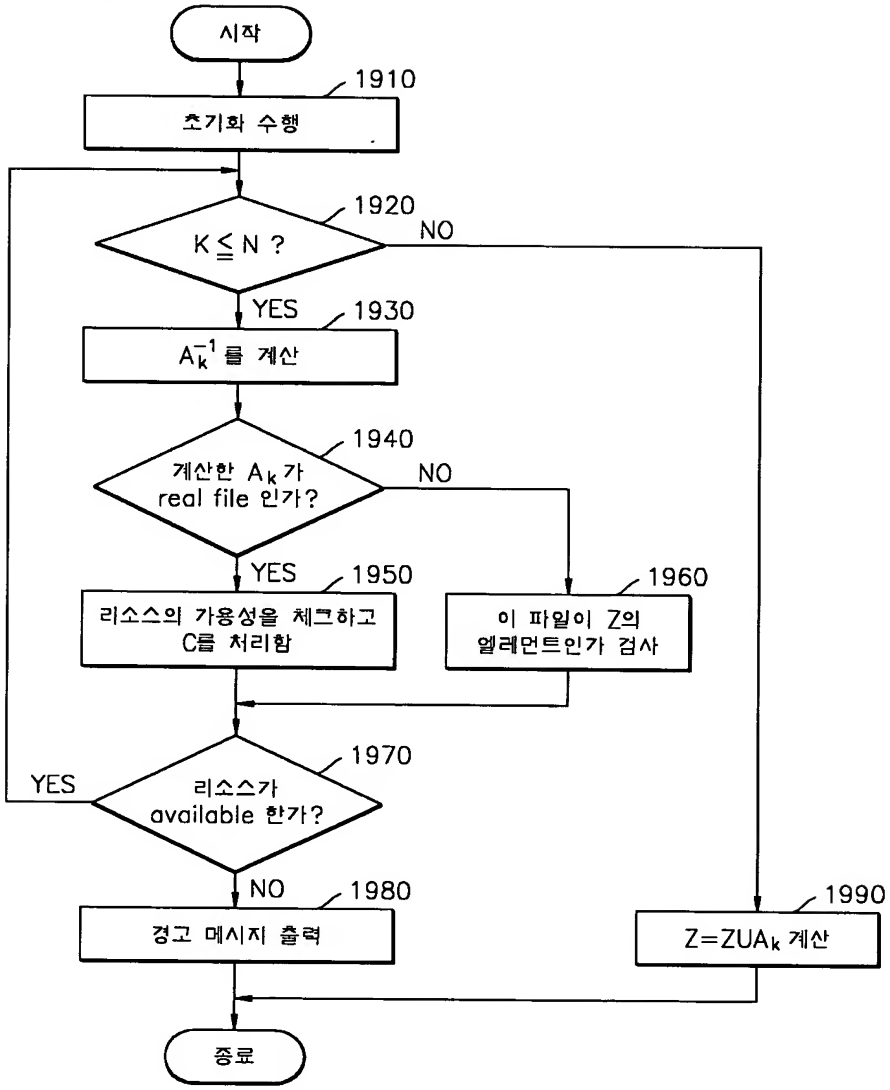
【도 17】

```
<VUNION/>
    <mstart=1>
    <mstop=31>
    <RESOURCE>
       <src="AAA.AVI"/>
       <mstart=1>
       <mstop=4/>
    </RESOURCE>
    <RESOURCE>
       <src="BBB.AVI"/>
       <mstart=0/>
       <mstop=23/>
       <start=4/>
    </RESOURCE>
    <RESOURCE>
       <src="AAA.AVI"/>
       <mstart=26>
       <mstop=31>
    </RESOURCE>
</VUNION>
```

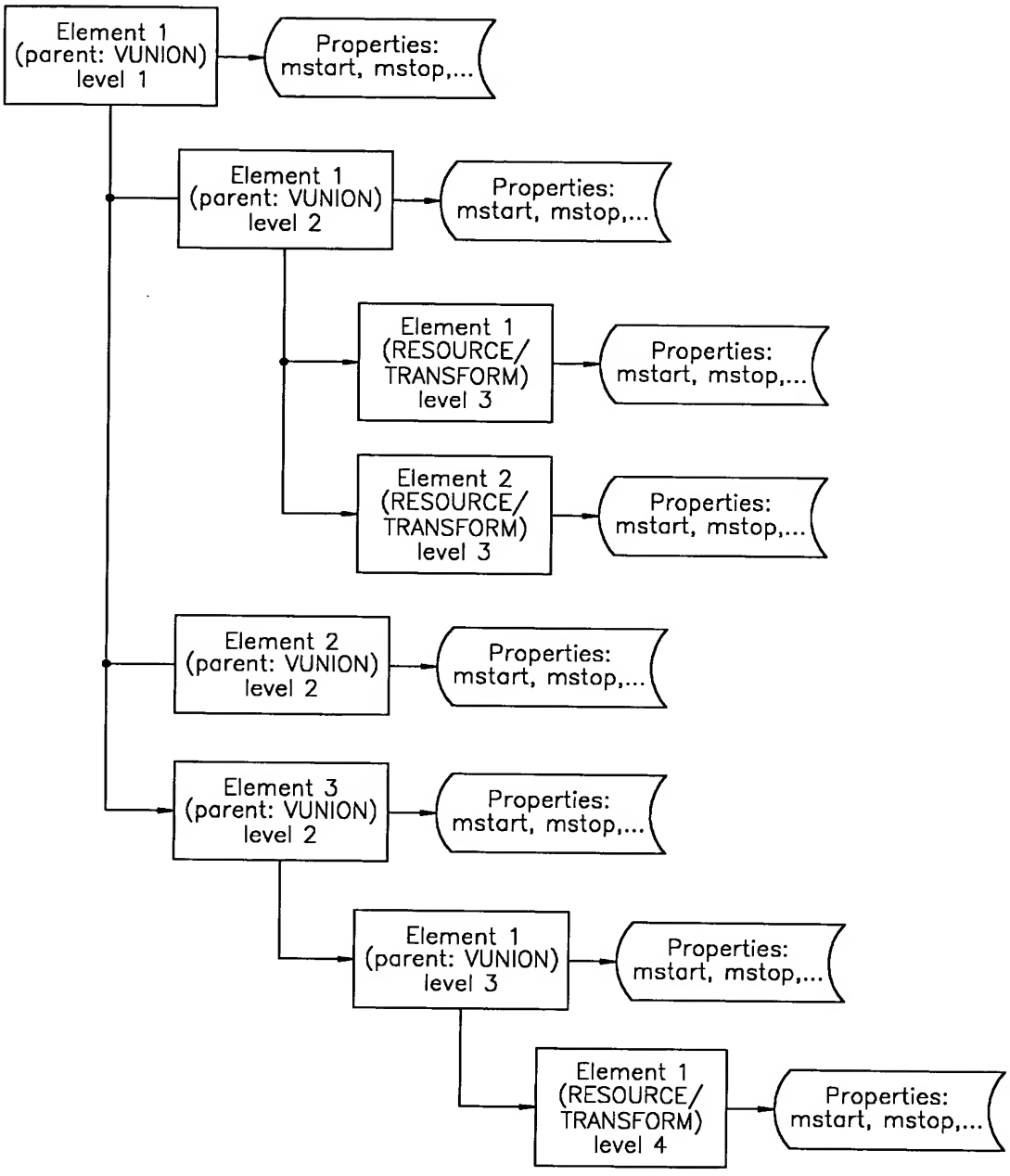
【도 18】



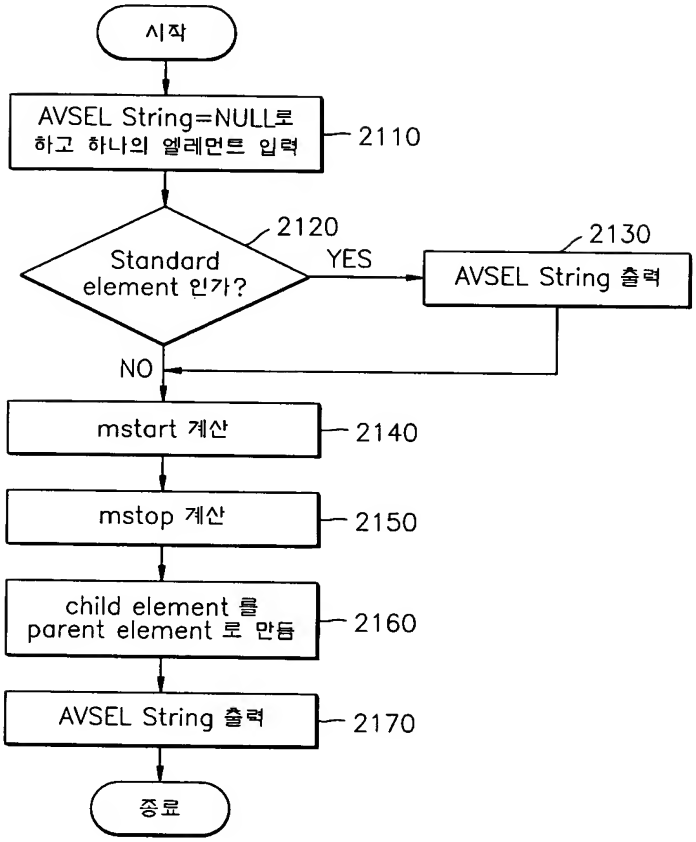
【도 19】


시작
1910
초기화 수행
1920
K≦N ?
NO
YES
1930
A_k^{-1} 를 계산
1940
계산한 A_k 가 real file 인가?
NO
YES
1950
리소스의 가용성을 체크하고 C를 처리함
1960
이 파일이 Z의 엘레먼트인가 검사
1970
리소스가 available 한가?
YES
NO
1980
경고 메시지 출력
1990
$Z=ZUA_k$ 계산
종료

【도 20】


Element 1 (parent: VUNION) level 1
Properties: mstart, mstop,...
Element 1 (parent: VUNION) level 2
Properties: mstart, mstop,...
Element 1 (RESOURCE/ TRANSFORM) level 3
Properties: mstart, mstop,...
Element 2 (RESOURCE/ TRANSFORM) level 3
Properties: mstart, mstop,...
Element 2 (parent: VUNION) level 2
Properties: mstart, mstop,...
Element 3 (parent: VUNION) level 2
Properties: mstart, mstop,...
Element 1 (parent: VUNION) level 3
Properties: mstart, mstop,...
Element 1 (RESOURCE/ TRANSFORM) level 4
Properties: mstart, mstop,...

【도 21】


시작
AVSEL String=NULL로 하고 하나의 엘레먼트 입력
2110
Standard element 인가?
2120
YES
2130
AVSEL String 출력
NO
mstart 계산
2140
mstop 계산
2150
child element 를 parent element 로 만듬
2160
AVSEL String 출력
2170
종료